跟龙哥学真AI

大模型微调实操--llama-factory

llama-factory环境安装

前置准备

英伟达显卡驱动更新地址

https://www.nvidia.cn/Download/index.aspx?lang=cn

cuda下载安装地址

https://developer.nvidia.com/cuda-12-2-0-download-archive/

pytorch下载安装地址

https://pytorch.org/get-started/previous-versions/

llama-factory项目和文档地址

https://github.com/hiyouga/LLaMA-Factory

https://llamafactory.readthedocs.io/zh-cn/latest/getting_started/installation.html

python环境下载地址

https://www.python.org/downloads/

miniconda下载地址

https://docs.anaconda.com/miniconda/

git下载地址

https://git-scm.com/downloads

硬件环境校验

显卡驱动和CUDA的安装

https://www.nvidia.cn/Download/index.aspx?lang=cn

https://developer.nvidia.com/cuda-12-2-0-download-archive/

使用以下命令做最简单的校验

```
nvidia-smi
```

打开 cmd 输入 `nvcc -V`，若出现类似内容则安装成功

软件环境准备

拉取LLaMA-Factory代码

没有git先安装一个git:https://git-scm.com/downloads

也可以直接下载代码解压缩

运行以下指令以安装 LLaMA-Factory 及其依赖:

```
git clone --depth 1 https://github.com/hiyouga/LLaMA-Factory.git
cd LLaMA-Factory
pip install -e ".[torch,metrics]"
```

创建虚拟环境

最好是先建一个虚环境来安装

比如用conda建立虚环境，不过要先安装conda或者miniconda

python环境下载地址：https://www.python.org/downloads/

miniconda下载地址：https://docs.anaconda.com/miniconda/

```
#conda创建  python=3.10版本的虚环境
conda create -n llama_factory python=3.10
#激活conda创建的名字叫llama_factory的虚环境
conda activate llama_factory


#torch安装
conda install pytorch==2.3.1 torchvision==0.18.1 torchaudio==2.3.1 pytorch-
cuda=12.1 -c pytorch -c nvidia

#龙哥抖音号：龙哥紫貂智能
```

备注：

如果 venv创建的虚环境，比如在LLaMA-Factory目录创建一个python310的子目录

```
#创建虚环境
python -m venv  python310

#激活虚环境
python310/Scripts/activate
```

量化环境

如果您想在 Windows 上启用量化 LoRA（QLoRA），请根据您的 CUDA 版本选择适当的 bitsandbytes] 发行版本

```
pip install https://github.com/jllllll/bitsandbytes-windows-webui/releases/download/wheels/bitsandbytes-0.41.2.post2-py3-none-win_amd64.whl
```

QLoRA最好安装cuda11.8以上的版本如12.1，特别是使用AWQ等量化算法的基础模型时，cuda11.8可能出现一些pytorch的错误

如果大模型使用awq量化需要安装autoawq模块

```
pip install autoawq
```

安装后使用以下命令做简单的正确性校验

```
import torch
torch.cuda.current_device()
torch.cuda.get_device_name(0)
torch.__version__
```

如果识别不到可用的GPU，则说明环境准备还有问题，需要先进行处理，才能往后进

那多大的模型用什么训练方式需要多大的GPU呢，可参考 https://github.com/hiyouga/LLaMA-Factory?tab=readme-ov-file#hardware-requirement

启动 LLaMA-Factory

同时对本库的基础安装做一下校验，输入以下命令获取训练相关的参数指导, 否则说明库还没有安装成功

llamafactory-cli命令在使用的python虚拟换的scripts目录下，正常激活虚拟目录如下命令就可以使用

```
llamafactory-cli train -h
```

```
#llamafactory-cli.exe在python虚环境的  scripts目录下

llamafactory-cli webui

#也可以是
CUDA_VISIBLE_DEVICES=0 llamafactory-cli webui
```

注意：目前webui版本只支持单机单卡和单机多卡，如果是多机多卡请使用命令行版本

如果要开启 gradio的share功能，或者修改端口号

```
CUDA_VISIBLE_DEVICES=0 GRADIO_SHARE=1 GRADIO_SERVER_PORT=7860 llamafactory-cli
webui
```

如图所示，上述的多个不同的大功能模块都通过不同的tab进行了整合，提供了一站式的操作体验。

手动下载模型

以Meta-Llama-3-8B-Instruct为例，通过huggingface 下载（可能需要先提交申请通过），当然我们一般就不去下载meta官方的原始模型

```
git clone https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

#下面的下载地址可以正常下载chat版本的
https://huggingface.co/shenzhi-wang/Llama3-8B-Chinese-Chat/tree/main
```

modelscope 下载（适合中国大陆网络环境）

```
git clone https://www.modelscope.cn/LLM-Research/Meta-Llama-3-8B-Instruct.git
```

代码下载模型

用modelscope库下载

```
#模型下载
from modelscope import snapshot_download
#linux系统
#local_dir = "/LLaMA-Factory/Qwen2-1.5B-Instruct"
#windows系统
model_dir = "F:/sotaAI/LLaMA-Factory/Qwen2-1.5B-Instruct"
model_dir = snapshot_download('qwen/Qwen2-1.5B-Instruct',local_dir=local_dir)
```

使用transformer来编写推理代码

```
import transformers
import torch

# 切换为你下载的模型文件目录，这里的demo是Qwen2-1.5B-Instruct
# 如果是其他模型，比如llama3，chatglm，请使用其对应的官方demo
#linux系统
#model_id = "/LLaMA-Factory/Qwen2-1.5B-Instruct"
#windows系统
model_id = "F:/sotaAI/LLaMA-Factory/Qwen2-1.5B-Instruct"

pipeline = transformers.pipeline(
```

```
    "text-generation",
    model=model_id,
    model_kwargs={"torch_dtype": torch.bfloat16},
    device_map="auto",
)

messages = [
    {"role": "system", "content": "你是一个电商客服，专业回答售后问题"},
    {"role": "user", "content": "你们这儿包邮吗?"},
]

prompt = pipeline.tokenizer.apply_chat_template(
        messages,
        tokenize=False,
        add_generation_prompt=True
)

#不同模型的eos_token_id不同，比如llama3,"<|eot_id|>"
terminators = [
    pipeline.tokenizer.eos_token_id,
    pipeline.tokenizer.convert_tokens_to_ids("<|im_end|>")
]

outputs = pipeline(
    prompt,
    max_new_tokens=256,
    eos_token_id=terminators,
    do_sample=True,
    temperature=0.6,
    top_p=0.9,
)
print(outputs[0]["generated_text"][len(prompt):])


#您好！感谢您对我们的关注。我们提供全国范围内免费快递服务，但是具体的运费信息需要根据您的收货地址和订单详情来计算，请您在下单时仔细核对并确认运费信息。如果您有任何疑问或需要帮助，欢迎随时联系我们。祝您购物愉快！

#龙哥抖音号：龙哥紫貂智能
```

微调数据集

偏好数据

sft微调一般用alpaca格式，dpo优化的偏好数据一般用sharegpt 格式

DPO优化偏好数据集

下面是DPO优化偏好数据集示例：

https://huggingface.co/datasets/hiyouga/DPO-En-Zh-20k/viewer/zh?row=5

HelpSteer2

英伟达开源的HelpSteer2

https://huggingface.co/datasets/nvidia/HelpSteer2

论文：https://arxiv.org/pdf/2406.08673

数据集注册

龙哥这儿拿甄嬛传里的语料当做自定义数据集举例

我们先下载 huanhuan.json数据集，是alpaca格式数据

可以在这儿下载：https://www.modelscope.cn/datasets/longgeai3x3/huanhuan-chat/files

微调过程

参数解析

微调命令

```
llamafactory-cli train \
    --stage sft \
    --do_train True \
    --model_name_or_path /data1/models/Llama3-8B-Chinese-Chat \
    --preprocessing_num_workers 16 \
    --finetuning_type lora \
    --template llama3 \
    --flash_attn auto \
    --dataset_dir /data1/workspaces/llama-factory/data/fiance-neixun \
    --dataset yinlian-sharegpt-neixun \
    --cutoff_len 1024 \
    --learning_rate 5e-05 \
    --num_train_epochs 3.0 \
    --max_samples 100000 \
    --per_device_train_batch_size 2 \
    --gradient_accumulation_steps 8 \
    --lr_scheduler_type cosine \
    --max_grad_norm 1.0 \
    --logging_steps 5 \
    --save_steps 100 \
    --warmup_steps 0 \
    --optim adamw_torch \
    --packing False \
    --report_to none \
    --output_dir saves/LLaMA3-8B-Chat/lora/train_2024-06-18-09-02-25 \
    --fp16 True \
    --plot_loss True \
    --ddp_timeout 180000000 \
    --include_num_input_tokens_seen True \
    --lora_rank 8 \
    --lora_alpha 16 \
    --lora_dropout 0 \
```

```
    --lora_target all \
    --deepspeed cache/ds_z3_config.json
```

也可以将参数安装格式保存为ymal文件，然后如下使用，具体格式可以参考根目录下的examples文件夹下的例子

```
llamafactory-cli train examples/train_lora/llama3_lora_sft.yaml
```

如果是windows下，在cmd窗口中，把命令中\和换行删掉，当成一行命名即可

```
llamafactory-cli train     --stage sft     --do_train True     --
model_name_or_path Qwen/Qwen2-7B-Instruct-AWQ     --preprocessing_num_workers 16
    --finetuning_type lora     --template qwen     --flash_attn auto     --
dataset_dir data     --dataset huanhuan_chat     --cutoff_len 1024     --
learning_rate 5e-05     --num_train_epochs 3.0     --max_samples 100000     --
per_device_train_batch_size 2     --gradient_accumulation_steps 8     --
lr_scheduler_type cosine     --max_grad_norm 1.0     --logging_steps 5     --
save_steps 100     --warmup_steps 0     --optim adamw_torch     --packing False
  --report_to none     --resume_from_checkpoint F:\sotaAI\LLaMA-
Factory\saves\Qwen2-7B-int4-Chat\lora\train_2024-08-17-15-56-58\checkpoint-500
  --output_dir saves\Qwen2-7B-int4-Chat\lora\train_2024-08-18-14-43-59     --fp16
True     --plot_loss True     --ddp_timeout 180000000     --
include_num_input_tokens_seen True     --quantization_bit 4     --
quantization_method bitsandbytes     --lora_rank 8     --lora_alpha 16     --
lora_dropout 0     --lora_target all
```

上面命令可以用来基于qwen2-7b训练huanhuan_chat的lora

中断继续训练

中断之后继续训练，可以使用下面命令，训练步数也会从保存的checkpoint处开始，比如checkpoint保存点是400步，但是在450步中断，会从400步开始继续

```
--resume_from_checkpoint
/workspace/checkpoint/codellama34b_5k_10epoch/checkpoint-4000
--output_dir new_dir
#--resume_lora_training #这个可以不设置

#如果不需要换output_dir，另外两条命令都不加，脚本会自动寻找最新的 checkpoint
--output_dir /workspace/checkpoint/codellama34b_5k_10epoch
```

当然使用命令训练，没有用webui看loss那么直观，需要加一个命令

```
--plot_loss  # 添加此参数以生成loss图
```

在训练结束后，loss图会保存在`--output_dir`指定的目录中

如果可以通过添加命令，从检查点开始继续训练，但训练集会从头开始训练，适合用新数据集继续训练

```
#lora的保存路径在llama-factory根目录下，如saves\Qwen2-7B-int4-Chat\lora\train_2024-07-17-15-56-58\checkpoint-500
--adapter_name_or_path    lora_save_patch
```

也可以在webui中指定 检查定路径，把路径复制进去

模型评估

大模型主流评测 benchmark

虽然大部分同学的主流需求是定制一个下游的垂直模型，但是在部分场景下，也可能有同学会使用本项目来做更高要求的模型训练，用于大模型刷榜单等，比如用于评测mmlu等任务。当然这类评测同样可以用于评估大模型二次微调之后，对于原来的通用知识的泛化能力是否有所下降。（因为一个好的微调，尽量是在具备垂直领域知识的同时，也保留了原始的通用能力）

在完成模型训练后，您可以通过 `llamafactory-cli eval examples/train_lora/llama3_lora_eval.yaml` 来评估模型效果。

配置示例文件 `examples/train_lora/llama3_lora_eval.yaml` 具体如下：

```
### examples/train_lora/llama3_lora_eval.yaml
### model
model_name_or_path: meta-llama/Meta-Llama-3-8B-Instruct
adapter_name_or_path: saves/llama3-8b/lora/sft # 可选项

### method
finetuning_type: lora

### dataset
task: mmlu_test
template: fewshot
lang: en
n_shot: 5

### output
save_dir: saves/llama3-8b/lora/eval

### eval
batch_size: 4
```

本脚本改编自 https://github.com/hiyouga/LLaMA-Factory/blob/main/examples/train_lora/llama3_lora_eval.yaml

如果是chat版本的模型

```
CUDA_VISIBLE_DEVICES=0 llamafactory-cli eval \   #eval表示评测
--model_name_or_path /llama3/Meta-Llama-3-8B-Instruct \   #基础模型路径
--template llama3 \   #提示词模版
--task mmlu_test \   #评测任务集
--lang en \          #语言
--n_shot 5 \         #0 shot,5 shot等
--batch_size 1       #评测是的batch size
```

windows下，评测命令实例：

```
llamafactory-cli eval  --model_name_or_path Qwen/Qwen2-7B-Instruct-AWQ --
adapter_name_or_path F:\sotaAI\LLaMA-Factory\saves\Qwen2-7B-int4-
Chat\lora\train_2024-08-18-14-43-59  --finetuning_type lora --template qwen  --
task cmmlu_test  --lang zh  --n_shot 3  --batch_size 1
```

大语言模型评估集

两个开源自动化评测项目：

https://github.com/open-compass/opencompass

https://github.com/EleutherAI/lm-evaluation-harness/tree/main

批量推理

环境准备

使用自动化的bleu和 rouge等常用的文本生成指标来做评估。指标计算会使用如下3个库，请先做一下pip安装

```
pip install jieba   #中文文本分词库
pip install rouge-chinese
pip install nltk   #自然语言处理工具包（Natural Language Toolkit）
```

参数解释

下面是一个 批量推理的命令例子

```
CUDA_VISIBLE_DEVICES=0 llamafactory-cli train \
    --stage sft \    #监督微调
    --do_predict \       #现在是预测模式
    --model_name_or_path /llama3/Meta-Llama-3-8B-Instruct \   #底模路径
    --adapter_name_or_path ./saves/LLaMA3-8B/lora/sft  \                #lora路径
    --eval_dataset alpaca_gpt4_zh,identity,adgen_local \     #评测数据集
    --dataset_dir ./data \     #数据集路径
```

```
    --template llama3 \       #提示词模版，比如llama3 ,qwen 和训练微调一样
    --finetuning_type lora \     #微调方式 lora
    --output_dir ./saves/LLaMA3-8B/lora/predict \     #评估预测输出文件夹
    --overwrite_cache \
    --overwrite_output_dir \
    --cutoff_len 1024 \       #提示词截断长度
    --preprocessing_num_workers 16 \     #预处理数据的线程数量
    --per_device_eval_batch_size 1 \    #每个设备评估时的batch size
    --max_samples 20 \      #每个数据集采样多少用于预测对比
    --predict_with_generate     #现在用于生成文本
```

推理示例

命令预览

```
llamafactory-cli train `
    --stage sft `
    --model_name_or_path Qwen/Qwen2-7B-Instruct-AWQ `
    --preprocessing_num_workers 16 `
    --finetuning_type lora `
    --quantization_method bitsandbytes `
    --template qwen `
    --flash_attn auto `
    --dataset_dir data `
    --eval_dataset huanhuan_chat,ruozhiba_gpt4 `
    --cutoff_len 1024 `
    --max_samples 100000 `
    --per_device_eval_batch_size 2 `
    --predict_with_generate True `
    --max_new_tokens 512 `
    --top_p 0.7 `
    --temperature 0.95 `
    --output_dir saves\Qwen2-7B-int4-Chat\lora\eval_2024-08-24-10-42-52 `
    --do_predict True `
    --adapter_name_or_path saves\Qwen2-7B-int4-Chat\lora\train_2024-08-18-14-43-
59 `
    --quantization_bit 4
```

windows下，龙哥的测试命令：

```
llamafactory-cli train     --stage sft     --model_name_or_path Qwen/Qwen2-7B-
Instruct-AWQ     --preprocessing_num_workers 16     --finetuning_type lora     --
quantization_method bitsandbytes     --template qwen     --flash_attn auto     --
dataset_dir data     --eval_dataset huanhuan_chat,ruozhiba_gpt4     --cutoff_len
1024     --max_samples 20    --per_device_eval_batch_size 2     --
predict_with_generate True     --max_new_tokens 512     --top_p 0.7     --
temperature 0.95     --output_dir saves\Qwen2-7B-int4-Chat\lora\eval_2024-08-24-
10-42-52  --do_predict True     --adapter_name_or_path saves\Qwen2-7B-int4-
Chat\lora\train_2024-08-18-14-43-59     --quantization_bit 4
```

模型部署

LoRA模型合并导出

也可以参考文档：

https://llamafactory.readthedocs.io/zh-cn/latest/getting_started/merge_lora.html

```
CUDA_VISIBLE_DEVICES=0 llamafactory-cli export \
    --model_name_or_path /llama3/Meta-Llama-3-8B-Instruct \
    --adapter_name_or_path ./saves/LLaMA3-8B/lora/sft  \
    --template llama3 \
    --finetuning_type lora \
    --export_dir megred-model-path \
    --export_size 2 \
    --export_device cpu \
    --export_legacy_format False
```

```
CUDA_VISIBLE_DEVICES=0 llamafactory-cli export \  #CUDA_VISIBLE_DEVICES是一个环境变量，用于指定程序使用的GPU设备编号。在这里，它设置为0，意味着告诉程序只使用第一张GPU卡（如果存在）
    --model_name_or_path /llama3/Meta-Llama-3-8B-Instruct \   #底模路径
    --adapter_name_or_path ./saves/LLaMA3-8B/lora/sft  \    #lora路径，量化导出时就不用lora路径啦
    --template llama3 \      #提示词模版
    --finetuning_type lora \      #微调方式lora
    --export_dir megred-model-path \   #导出路径
    --export_size 2 \          #导出每个分文件大小，2表示2G，比如一个4B权重的float16的模型，模型权重8G，就会分成4个 2G文件保存
    --export_device cpu \    #这儿的export_device cpu 指的是导出是这个动作使用cpu，而不是模型后面是用cpu还是gpu运行，模型权重是没有cpu和gpu之分的
    --export_legacy_format False  #是否使用旧格式导出，新格式默认是safetensors，旧格式就是pt，bin等格式
```

还有一些参数可以参考：

```
model_name_or_path: 预训练模型的名称或路径
template: 模型模板
export_dir: 导出路径
export_quantization_bit: 量化位数，全精度导出时，不用填写
export_quantization_dataset: 量化校准数据集
export_size: 最大导出模型文件大小，如果模型权重大小比较大，就会分成多个文件导出
export_device: 导出设备
export_legacy_format: 是否使用旧格式导出
```

导出GGUF

开源项目llama.cpp提供的有模型格式转换工具

项目地址：https://github.com/ggerganov/llama.cpp

#####

安装gguf库

如果直接 pip 安装 gguf，不是最新版本，和最新的转换脚本会不兼容，可以直接从源码安装llama.cpp

```
#conda创建  python=3.10版本的虚环境
conda create -n llama_cpp python=3.10
#激活conda创建的名字叫llama_cpp的虚环境
conda activate llama_cpp

#torch安装，cuda安装,等等
conda install pytorch==2.3.1 torchvision==0.18.1 torchaudio==2.3.1 pytorch-
cuda=12.1 -c pytorch -c nvidia

#拉取代码
git clone https://github.com/ggerganov/llama.cpp.git
cd llama.cpp
pip install --editable .
```

如有必要按照模型训练环境准备章节 建立一个llama.cpp需要的python虚拟环境

格式转换

返回 llama.cpp 项目根目录，会有一个官方提供的 convert-hf-to-gguf.py 脚本，用于完成huggingface格式到gguf格式的转换

```
#在llama.cpp根目录
python convert_hf_to_gguf.py F:\sotaAI\LLaMA-Factory\saves\export     #需要转换的模
型路径
```

ollama安装

我们以ollama为例，下载地址：https://ollama.com/

ollama是go语言开发的开源项目，github地址：https://github.com/ollama/ollama

ollama文档参考：https://github.com/ollama/ollama/tree/main/docs

ollama支持的是gguf文件格式，如果是其他文件格式需要转换成gguf文件格式

linux安装

一般线上gpu算力服务器，驱动都会已经安装好

虽然 AMD 已将 `amdgpu` 驱动程序上游贡献给官方 Linux 内核源代码，但该版本较旧，可能不支持所有 ROCm 功能。我们建议您从 https://www.amd.com/en/support/linux-drivers 安装最新驱动程序，以获得对您 Radeon GPU 的最佳支持。

```
#运行以下一行命令来安装 Ollama
curl -fsSL https://ollama.com/install.sh | sh


#或者手动下面命令安装

sudo curl -L https://ollama.com/download/ollama-linux-amd64 -o /usr/bin/ollama
sudo chmod +x /usr/bin/ollama

#将 Ollama 添加为启动服务（推荐）
#为 Ollama 创建一个用户
sudo useradd -r -s /bin/false -m -d /usr/share/ollama ollama

#在 /etc/systemd/system/ollama.service 中创建一个服务文件，内如如下，截止    启动服务   前

[Unit]
Description=Ollama Service
After=network-online.target

[Service]
ExecStart=/usr/bin/ollama serve
User=ollama
Group=ollama
Restart=always
RestartSec=3

[Install]
WantedBy=default.target




#然后启动服务
sudo systemctl daemon-reload
sudo systemctl enable ollama


#启动 Ollama
sudo systemctl start ollama

#移除 ollama 服务
sudo systemctl stop ollama
sudo systemctl disable ollama
sudo rm /etc/systemd/system/ollama.service
```

windows安装

不再需要 WSL！

Ollama 现在已作为 Windows 原生应用程序运行，包括对 NVIDIA 和 AMD Radeon GPU 的支持

下载地址：https://ollama.com/download，选择windows系统直接安装启动

ollama默认会安装在C:\Users\文件夹下，一般会有下面三个位置

```
#程序文件目录
C:\Users\Administrator\AppData\Local\Programs\Ollama
#日志文件夹
C:\Users\Administrator\AppData\Local\Ollama

#模型和数据文件夹，如果设置了OLLAMA_MODELS环境变量，模型会下载注册到到指定的文件夹下
C:\Users\Administrator\.ollama
```

系统要求

Windows 10 或更新版本，家庭版或专业版

如果您有 NVIDIA 显卡，需要 NVIDIA 452.39 或更新版本的驱动程序:https://www.nvidia.cn/Download/index.aspx?lang=cn

如果您有 Radeon 显卡，需要 AMD Radeon 驱动程序 https://www.amd.com/en/support

启动

安装 Ollama Windows 预览版后，Ollama 将在后台运行，`ollama` 命令行可在 `cmd`、`powershell` 或您喜爱的终端应用中使用

后台使用命令，管理ollama

```
#启动ollama服务
./ollama serve

#加载gemma2模型的2b版本，这个最好指定一下版本，不然模型太大，下载费时间
#如果模型没有下载，会自动下载到下面 OLLAMA_MODELS 这个环境变量的目录下，没有设置的话
./ollama run gemma2:2b
```

环境变量

下面两个环境变量，可以设置模型目录和ollama api服务的url

```
OLLAMA_MODELS     d:/ollama/models      #可以自定义模型目录，不然默认安装，模型目录在C:\Users\文件夹下

OLLAMA_BASE_URL    http://127.0.0.1:11434      #可以自定义端口
```

ollama命令介绍

官方支持模型

ollama支持的模型列表：https://ollama.com/library

对于ollama支持的模型可以用下面命令来部署

```
ollama pull llama3.1    #下载模型
ollama rm llama3.1      #删除模型
ollama show llama3.1  #显示模型信息
ollama list      #列出下载好的模型
```

自定义模型

ollama支持的是gguf文件格式，如果是其他文件格式需要转换成gguf文件格式

要新建一个 Modelfile文件，文件内容如下，该文件假设路径是 d:/ollama/longgemf

```
FROM llama3.1      #官方支持的模型
#FROM d:/ollama/llama3-huanhuan.gguf  #必须    自定义支持的模型，路径加模型文件名

#不是必须 温度参数，参数越大，分布曲线压的越平
PARAMETER temperature 1

#不是必须   设置系统提示词，
SYSTEM """
You are Mario from Super Mario Bros. Answer as Mario, the assistant, only.
"""
```

需要说明的是，因为LLaMA3的实现本身相对规范，所以Modelfile 的编写相对很简洁，反之对于其他一些模型，可能还需要在里面自主完成template prompt格式的注册，否则问答时会出现答非所问的情况

注册模型

注册了一个叫huanhuan的模型，用下面命令：

```
ollama create huanhuan -f d:/ollama/longgemf    # -f 后面是自定义的 modelfile文件
```

命令聊天示例

```
ollama run huanhuan      #运行模型就可以输入命令
>>> 你是谁
我是甄嬛，家父是大理寺少卿甄远道
```

龙哥实验例子：

```
#设置一下模型目录环境变量
OLLAMA_MODELS=F:\sotaAI\ollama-openwebui\llm_models

#longgemf这个 modelfile文件内容
FROM F:\sotaAI\LLaMA-Factory\saves\export\Qwen2-0.5B-F16.gguf

#运行命令注册huanhuan
ollama create huanhuan -f F:\sotaAI\ollama-openwebui\llm_models\longgemf

#使用huanhuan进行推理
ollama run huanhuan
```

open-webui 本地模型部署ui项目

如果需要本地电脑部署一个大模型带ui界面的项目，方便管理模型，可以管理文档资料，方便和gpt一样聊天，可以安装一个open-webui项目

open-webui是一个开源的本地模型推理的webui项目，后端和ollama兼容

项目下载地址：https://github.com/open-webui/open-webui

API调用服务

llama-factory的api服务

训练好后，可能部分同学会想将模型的能力形成一个可访问的服务，通过API 来调用，接入到langchian或者其他下游业务中，项目也自带了这部分能力。

API 实现的标准是参考了OpenAI的相关接口协议，基于uvicorn服务框架进行开发， 使用如下的方式启动

```
CUDA_VISIBLE_DEVICES=0 API_PORT=8000 llamafactory-cli api \
    --model_name_or_path /llama3/Meta-Llama-3-8B-Instruct \
    --adapter_name_or_path ./saves/LLaMA3-8B/lora/sft \
    --template llama3 \
    --finetuning_type lora
```

如果要加速推理可以是用vllm推理后端，不过vllm只支持linux系统，不支持windows系统

vllm开源项目地址：https://github.com/vllm-project/vllm

项目也支持了基于vllm 的推理后端，但是这里由于一些限制，需要提前将LoRA 模型进行merge，使用merge后的完整版模型目录或者训练前的模型原始目录都可。

```
CUDA_VISIBLE_DEVICES=0 API_PORT=8000 llamafactory-cli api \
    --model_name_or_path megred-model-path \
    --template llama3 \
    --infer_backend vllm \
    --vllm_enforce_eager
```

服务启动后，即可按照openai 的API 进行远程访问，主要的区别就是替换 其中的base_url，指向所部署的机器url和端口号即可。

```
import os
from openai import OpenAI
from transformers.utils.versions import require_version

require_version("openai>=1.5.0", "To fix: pip install openai>=1.5.0")

if __name__ == '__main__':
    # change to your custom port
    port = 8000
    client = OpenAI(
        api_key="0",
        base_url="http://localhost:{}/v1".format(os.environ.get("API_PORT",
8000)),
    )
    messages = []
    messages.append({"role": "user", "content": "hello, where is USA"})
    result = client.chat.completions.create(messages=messages, model="test")
    print(result.choices[0].message)
```

ollama的api服务

ollama api

启动ollama服务后，就可以通过api调用来进行推理，api url可以通过环境变量 OLLAMA_BASE_URL 来指定，包括端口号，默认就是

```
http://127.0.0.1:11434
```

比如支持 流式生成或者非流式生成，通过post请求生成聊天内容

```
POST /api/generate
POST /api/chat
```

详细api文档可以参考

https://github.com/ollama/ollama/blob/main/docs/api.md

openai兼容api

支持openai兼容的api接口，而且国内的大部门闭源模型也支持openai兼容的api，这样不用重复开发多套接口，直接在原有模型支持的基础上，换模型和api url地址，就可以支持ollama模型推理

https://github.com/ollama/ollama/blob/main/docs/openai.md

python例子代码

```python
from openai import OpenAI

client = OpenAI(
    base_url='http://localhost:11434/v1/',

    # required but ignored
    api_key='ollama',
)

#用llama3聊天
chat_completion = client.chat.completions.create(
    messages=[
        {
            'role': 'user',
            'content': 'Say this is a test',
        }
    ],
    model='llama3',
)

#用llava多模态，输入一张图片做总结
response = client.chat.completions.create(
    model="llava",
    messages=[
        {
            "role": "user",
            "content": [
                {"type": "text", "text": "What's in this image?"},
                {
                    "type": "image_url",
```

```
                    "image_url":
"iVBORw0KGgoAAAANSUhEUgAAAG0AAABmCAYAAADBPx+VAAAACXBIWXMAAAsTAAALEwEAmpwYAAAAAXNS
R0IArs4c6QAAAARnQU1BAACxjwv8YQUAAA3VSURBVHgB7Z27r0zdG8fX743i1bi1ikMoFMQloXRpKFFIq
I7LH4BEQ+NWIkjQuSWCRIEoULk0gsK1kCBIOIhrQVT7tz/7zZo888yz1r7MnDl7z5xvsjkzs2fP3uu71n
Nfa7lkAsm7d++Sffv2JbNmzUqcc8m0adOSzZs3Z+/XES4ZckAWJEGWPiCxjsQNLWmQsWjRIpMseaxcuTK
pG/7HP27I8P79e7dq1ars/yL4/v27SOejqwv+cUOGEGGpKHR37tzJCEpHV9tnT58+dXXCJDdECBE2Ojrq
jh071hpNECjx4cMHVycM1Uhbv359B2F79+51586daxN/+pyRkRFXKyRDAqxEp4yMlDDzXG1NPnnyJKkTh
oK0VFd1ELZu3TrzXKxKfW7dMBQ6bcuWLW2v0VlHjx41z717927ba22U9APcw7Nnz1oGEPeL3m3p2mTAYY
nFmMOMXybPPXv2bNIPpFZr1NHn4HMw0KRBjg9NuRw95s8PEcz/6DZELQd/09C9QGq5RsmSRybqkwHGjh0
7OsJSsYYm3ijPpyHzoiacg35MLdDSIS/O1yM778jOTwYUkKNHWUzUWaOsylE0OMyI0fcnOwIdjvtNdW/H
ZwNLGg+sR1kMepSNJXmIwxBZiG8tDTpEZzKg0GItNsosY8USkxDhDORinuiko2gfL/RbiD2LZAjU9zKQJ
j8RDR0vJBR1/Phx9+PHj9Z7REF4nTZkxzX4LCXHrV271qXkBAPGfP/atWvu/PnzHe4C97F48eIsRLZ9+3
a3f/9+87dwP1JxaF7/3r17ba+5l4EcaVoOlj3SBq5kGTJSQmLWMjgYNei2GPT1MuMqGTDEFHzeQSP2wi/
jGnkmPJ/nhccs44jvDAxpVcxnq0F6eT8h4ni/iIWpR5lPyA6ETkNXoSukvpJAD3AsXLiwpZs49+fPn5ke
4j10TqYvegSfnOOnafC+Tv9ooA/JPkgQysqQNBzagXY55nO/oa1F7qvIPWkRL12WRpMWUvpVDYmxAPehx
WSe8ZEXL2OsadYIozfmNch4QJPAfeJgW3rNsnzphBKNJM2KKODo1rVOMRYik5ETy3ix4qWNI81qAAiriz
gMIc+yhTytxOJWZuNIO3qsrgWlGtwjoS9XwgUhWGyhUaRZZQNNIEwCiXD16tXcAHUs79coOvSD8rrJCIW
98pzvxpAWyyo3HYwqSO+HOBjStClcZJT5coMm6D2LOF8TolGJtK9fvyZpyiC5ePFi9nc/oJU4eiEPOjVo
AnHa9wyJycITMP78+eMeP37sXrx44d6+fdt6f82aNdkx1pg9e3Zb5W+RSRE+n+VjksQWifvVaTKFhn5O8
my63K8Qabdv33b379/PiAP//vuvW7BggZszZO72/+TJk91YgkafPn166zXB1rQHFvouAWHq9z3SEevSUe
rqCn2/dDCeta2jxYbr69evk4MHDyY7d+7MjhMnTiTPnz9Pfv/+nfQT2ggpO2dMF8cghuoM7Ygj5iWCqRl
GFml0QC/ftGmTmzt3rmsaKDsgBSPh0/8yPeLLBihLkOKJc0jp8H8vUzcxIA1k6QJ/c78tWEyj5P3o4u9+
jywNPdJi5rAH9x0KHcl4Hg57OeQp3+vHXGyrmEeigzQsQsjavXt38ujRo44LQuDDhw+TW7duRS1HGgMxh
NXHgflaNTOsHyKvHK5Ijo2jbFjJBQK9YwFd6RVMzfgRBmEfP37suBBm/p49e1qjEP2mwTViNRo0VJWH1d
eMXcNK08uUjVUu7s/zRaL+oLNxz1bpANco4npUgX4G2eFbpDFyQoQxojBCpEGSytmOH8qrH5Q9vuzD6of
QylkCUmh8DBAr+q8JCyVNtWQIidKQE9wNtLSQnS4jDSsxNHogzFuQBw4cyM61UKVsjfr3ooBkPSqqQHes
UPWVtzi9/vQi1T+rJj7WiTz4Pt/l3LxUkr5P2VYZaZ4URpsE+st/dujQoaBBYokbrz/8TJNQYLSonrPS9
kUaSkPeZyj1AWSj+d+VBoy1pIWVNed8POLl/ee5HdGRhrHhR5GGN0r4LGZBaj8oFDJitBTJzIZgFcmU0Y
8ytWMZMzJOaXUSrUs5RxKnrxmbb5YXO9VGUhtpXldhEUogFr3IzIsvlpmdosVcGVGXFWp2oU9kLFL3dEk
Sz6NHEY1sjSRdIuDFWEhd8KxFqsRi1uM/nz9/zpxnwlESONdg6dKlbsaMGS4EHFHtjFIDHwKOo46l4TxS
uxgDzi+rE2jg+BaFruOX4HXa0Nnf1lwAPufZeF8/r6zD97WK2qFnGjBxTw5qNGPxT+5T/r7/7RawFC3j4
vTpO9koCxkeHjqbHJqArmH5UrFKKksnxrK7FuRIs8STfBZv+luugXZ2pR/pP9Ois4z+TiMzUUkUjDOiEi
1fzX8GmXyuxUBRcaUfykVOYZnlJGKQpOiGB76x5GeWkWWJc3mOrK6S7xdND+W5N6XyaRgtWJFe13GkaZn
KOsYqGdOVVVbGupsyA/l7emTLHi7vwTdirNEtOqxnzAvBFcnQF16xh/TMpUuXHDowhlA9vQVraQhkudRd
zOnK+O4ZSP3DUhVSP61YsaLtd/ks7ZgtPcXqPqEafHkdqa84X6aCeL7YWlv6edGFHb+ZFICPlljHhgObK
uk0CSvVznWsotRu433alNdFrqG45ejoaPCaUkWERpLXjzFL2Rpllp7PJU2a/v7Ab8NO5/9t27Z16KUqoF
GsxnI9EosS2niSYg9SpU6B4JgTrvVW1flt1sT+OADIJU2maXzcUTraGCRaL1Wp9rUMk16PMom8QhruxzV
ZIegJjFU7LLCePfS8uaQdPny4jTTLOdbee5mYokQsXTIWNY46kuMbnt8Kmec+LGWtOVIl9cT1rCBOV8Wq
kjAsRwta93TbwNYoGKsUSChN44lgBNCoHLHzquYKrU6qZ8lolCINORh6cPOQ3U6I6IXILYOQI513hJaSK
AorFpuHXJNfVlpRtmYBk1Su1obZr5dnKAO+L1OHrj3WZW+E3qh6IszE37F6EB+68mGpvKm4eb9bFrlzro
k7fvrOKfv727dvWRmdVTJHwOqiiCUSZ6wCK+7XL/AcsgNyL74DQQ73Osv78Su7+t/A36MdYOsW5o4Oahs
lXr58aZ5HtZB8GH64m9EmMZ7FpYw4T6QnrZfgenrhFxaSiSGXtPnz57e9TkNZLvTjeqhr734CNtrK41L4
OsUQckmj1lGKQOrC37x544r8eNXRpnVE3ZZY7zXo8NomiO0ZUCj2uHz58rbXoZ6gc0uA+F6ZeKS/jhRDU
q8MKrTho9fEkihMmhxtBI1DxKFY9XLpVcSkfoi8JGnToZO5sU5aiDQIW716ddt7ZLYtMQlhECdBGXZZMW
ldY5BHm5xgAroWj4C0hbYkSc/jBmggIrXJWlZM6pSETsEPGqZOndr2uuuR5rF169a2HoHPdurUKZM4CO1
WTPqaDaAd+GFGKdIQkxAn9RuEWcTRyN2KSUgiSgF5aWzPTeA/lN5rZubMmR2bE4SIC4nJoltgAV/dVefZ
m72AtctUCJU2CMJ327hxY9t7EHbkyJFseq+EJSY16RPo3Dkq1kkr7+q0bNmyDuLQcZBEPYmHVdOBiJyIl
rRDq41YPWfXOxUysi5fvtyaj+2BpcnsUV/oSoEMOk2CQGlr4ckhBwaetBhjCwHOZHtJROPJkyc7UjcYLD
jmrH7ADTEBXFfOYmB0k9oYBOjJ8b4aOYSe7QkKcYhFlq3QYLQhSidNmtS2RATwy8YOM3EQJsUjKiaWZ+v
ZToUQgzhkHXudb/PW5YMHD9yZM2faPsMwoc7RciYJXbGuBqJ1UIGKKLv915jsvgtJxCZDubdXr165mzdv
tr1Hz5LONA8jrUwKPqsmVesKa49S3Q4WxmRPUEYdTjgiUcfUwLx589ySJUva3oMkP6IYddq6HMS4o55xB
JBUeRjzfa4Zdeg56QZ43LhxoyPo7Lf1kNt7oO8wWAbNwaYjIv5lhyS7kRf96dvm5Jah8vfvX3flyhX35c
uX6HfzFHOToS1H4BenCaHvO8pr8iDuwoUL7tevX+b5ZdbBair0xkFIlFDlW4ZknEClsp/TzXyAKVOmmHW
FVSbDNw1l1+4f90U6IY/q4v27dpnE9bJ+v87QEydjqx/UamVVPRG+mwkNTYN+9tjkwzEx+atCm/X9WvWt
DtAb68Wy9LXa1UmvCDDIpPkyOQ5ZwSzJ4jMrvFcr0rSjOUh+GcT4LSg5ugkW1IoO/SCDQBojh0hPlaJda
h+tkVYrnTZowP8iq1F1TgMBBauufyB33x1v+NWFYmT5KmppgHC+NkAgbmRkpD3yn9QIseXymoTQFGQmIO
```

```
KTxiZIWpvAatenVqRVXf2nTrAWMsPnKrMZHz6bJq5jvce6QK8J1cQNgKxlJapMPdZSR64/UivS9NztpkV
EdKcrs5alhhWP9NeqlfWopzhZScI6QxseegZRGeg5a8C3Re1Mfl1ScP36ddcUaMuv24iOJtz7sbUjTS4q
BvKmstYJoUauiuD3k5qhyr7QdUHMeCgLa1Ear9NquemdXgmum4fvJ6w1lqsuDhNrg1qSpleJK7K3TFOQ2
jSd94uSZ6OkK1e3qyVpQK6PVWXp2/FC3mp6jBhKKOiY2h3gtUV64TWM6wDETRPLDfSakXmH3w8g9Jlug8
ZtTt4kVFOkLUYYmCCtD/DrQ5YhMGbA9L3ucdjhOy8kOHW5gU/VEEmJTcL4Pz/f7mgoAbYkAAAAAElFTkS
uQmCC",
                },
            ],
        }
    ],
    max_tokens=300,
)

completion = client.completions.create(
    model="llama3",
    prompt="Say this is a test",
)

list_completion = client.models.list()

model = client.models.retrieve("llama3")

#用all-minilm模型做embeddings
embeddings = client.embeddings.create(
    model="all-minilm",
    input=["why is the sky blue?", "why is the grass green?"],
)
```